Amendments to the Claims

The pending claims are presented below.

1. (Cancelled)

2. (Cancelled)

3. (Cancelled)

4. (Previously Presented) A front-end system for a processor, comprising:

an instruction cache system;

an extended block cache system, comprising:

a fill unit coupled to the instruction cache system,

a block cache,

a block predictor to store masks associated with complex blocks, the masks distinguishing block prefixes from each other; and

a selector coupled to the output of the instruction cache system and to an output of the block cache.

5. (Previously Presented) The front-end system of claim 4, wherein the extended block cache system further comprises a block predictor coupled to the fill unit and the block cache.

6. (Previously Presented) The front-end system of claim 4, wherein the block cache is to store blocks having a multiple-entry, single exit architecture.

7. (Previously Presented) The front-end system of claim 4, wherein the block cache is to store complex blocks having multiple independent prefixes and a common suffix.

8. (Cancelled)

9. (Original) A method of managing extended blocks, comprising:

predicting an address of a terminal instruction of an extended block to be used,

determining whether the predicted address matches an address of a terminal instruction of a previously created extended block, and

selecting one of the extended block in the event of a match.

10. (Original) The method of claim 9, further comprising creating a new extended block when there is no match.

11. (Original) The method of claim 10, wherein the creating comprises:

receiving new instructions until a terminal condition occurs,

assembling the new instructions into an extended block,

determining whether an address of a terminal instruction in the new block matches an address of a terminal instruction of a pre-existing block, and

unless a match occurs, storing the new block in a memory.

12. (Original) The method of claim 11, wherein the storing comprises, when an older block causes a match, storing the new block over the old block in a memory if the old block is subsumed within the new block.

13. (Original) The method of claim 11, wherein the storing comprises, when an older block causes a match, dropping the new block if the new block is subsumed within the older block.

14. (Original) The method of claim 11, wherein the storing comprises, when an older block causes a match, creating a complex block if the new block and the older block share a common suffix but include different prefixes.

15. (Previously Presented) The method of claim 9, further comprising outputting instructions of the selected extended block for execution.

16. (Cancelled)

17. (Previously Presented) A processing engine, comprising:

a front end stage storing blocks of instructions in a multiple-entry, single exit architecture when considered according to program flow, and

an execution unit in communication with the front end stage,

wherein the front-end stage includes

an instruction cache system,

an extended block cache system, including

a fill unit provided in communication with the instruction cache system,

a block cache, and

a selector coupled to the output of the instruction cache system and to an output of the block cache.

18. (Original) The processing engine of claim 17, wherein the block cache is to store the multiple-entry, single exit traces.

19. (Previously Presented) The processing engine of claim 17, wherein the extended block cache system further comprises a block predictor coupled to the fill unit and the block cache.

20. – 21. (Cancelled)

22. (Cancelled)

23. (Cancelled)

24. (Cancelled)

25. (Cancelled)

26. (Cancelled)

27. (Previously Presented) A memory comprising storage for a plurality of traces and means for indexing the traces by an address of a last instruction therein according to program flow, wherein the traces include a plurality of instructions assembled according to program flow and at least one trace includes at least three segments of executable instructions in which, when considered according to program flow, first and second segments are mutually exclusive of each other and lead into the third segment.

28 – 38. (Cancelled)

39. (Cancelled)

40. (Cancelled)

41. (Cancelled)

42. (Cancelled)

43. (Cancelled)